BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Hasil Penelitian

Berdasarkan metodelogi yang telah dirancang untuk membandingkan tingkat akurasi yang tertinggi dalam memprediksi resiko terkena penyakit jantung dengan menggunakan dua algoritma yaitu *random forest* dan *decision tree* maka hasil yang didapatkan adalah sebagai berikut:

4.1.1 *Data Collection*

Langkah pertama dalam proses data mining adalah pengumpulan data. Data yang telah dikumpulkan melalui platform repositori dari *kaggle* berjumlah 1888 set data yang terdiri dari 14 *column*, antara lain: *age, sex, cp, tretbps, chol, fbs, restecg, thalachh, exang, oldpeak, slope, ca, thal,* dan *target*, resiko terkena dan tidak terkena penyakit jantung dalam format csv.

	age	sex	cp	trestbps	chol	fbs	restecg	thalachh	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Gambar 4.1 Sampel Data

4.1.2 *Expolatory Data Analysis (EDA)*

a) Tipe Data

Melihat dan mengecek tipe data dari masing-masing variabel karena ini dapat mempengaruhi kualitas data dan dapat mengetahui proses selanjutnya.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1888 entries, 0 to 1887
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1888 non-null   int64  
 1   sex       1888 non-null   int64  
 2   cp        1888 non-null   int64  
 3   trestbps  1888 non-null   int64  
 4   chol      1888 non-null   int64  
 5   fbs       1888 non-null   int64  
 6   restecg   1888 non-null   int64  
 7   thalachh  1888 non-null   int64  
 8   exang     1888 non-null   int64  
 9   oldpeak   1888 non-null   float64
 10  slope     1888 non-null   int64  
 11  ca        1888 non-null   int64  
 12  thal      1888 non-null   int64  
 13  target    1888 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 206.6 KB
```

Gambar 4.2 Hasil Dari Cek Tipe Data

Setelah dilakukan pengecekan tipe data bahwa data sudah sesuai untuk dilakukan analisis tahap selanjutnya karena sudah bertipe angka dan hanya kolom *oldpeak* bertipe angka pecahan desimal.

b) *Missing Value*

Pada tahap ini dilakukan pemeriksaan pada dataset yang bertujuan mengetahui nilai yang rumpang atau *missing value* untuk menghindari *overfitting* pada model, data yang rumpang sangat sensitif terhadap performa model. Oleh karena itu, harus dilakukan pemeriksaan missing value. Pengecekan *missing value* pada setiap variabel sangat diperlukan karena akan dapat mempengaruhi hasil dari training, sehingga jika ada *missing value* pada variabel harus segera diatasi dan ditangani dengan menghapusnya atau drop kolom. Hasil pengecekan pada setiap variabel adalah:

	0
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	0
restecg	0
thalachh	0
exang	0
oldpeak	0
slope	0
ca	0
thal	0
target	0

dtype: int64

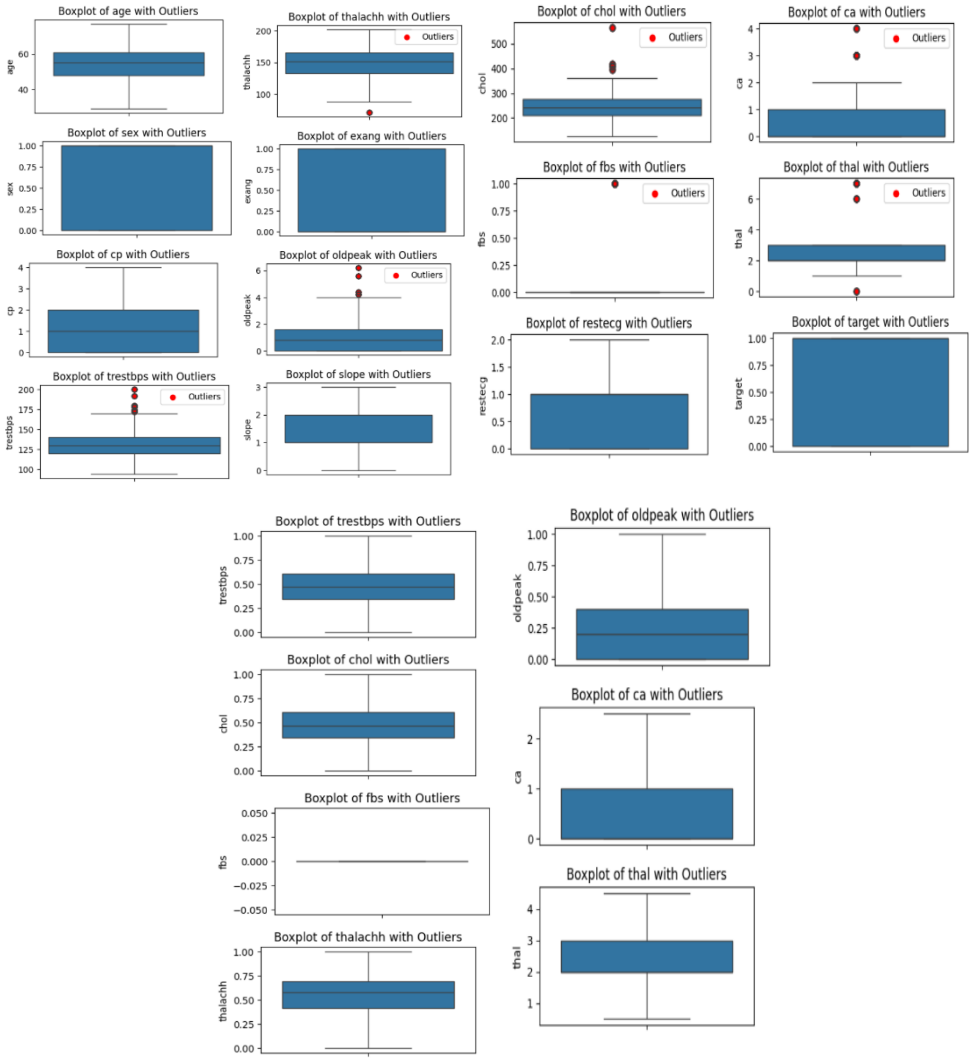
Gambar 4.3 *Missing Value*

Setelah dilakukanya pengecekan *missing value*, ternyata tidak terdapat adanya *missing value* atau tidak terdeteksi berjumlah 0 atau tidak ada sama sekali, sehingga data masih berjumlah 1888 dapat dilakukan proses selanjutnya.

c) *Outlier*

Pada tahap ini, pemeriksaan dataset adanya kemungkinan terdapat data *Outlier*. Data *outlier* dapat mempengaruhi performa model karena akan mengakibatkan terjadinya overfitting karena data tersebut bukan data yang termasuk dalam sebenarnya.

Berdasarkan gambar 4.4 dalam tahap itu pemeriksaan outlier menggunakan fitur bloxpot dan IQR *(Inter Quartile Range)* pada data *Trestbps*, *Oldpeak*, *Chol, Fbs, Restecg, Ca*, dan *Thal* terdapat outlier.



Gambar 4.4 *Outlier*

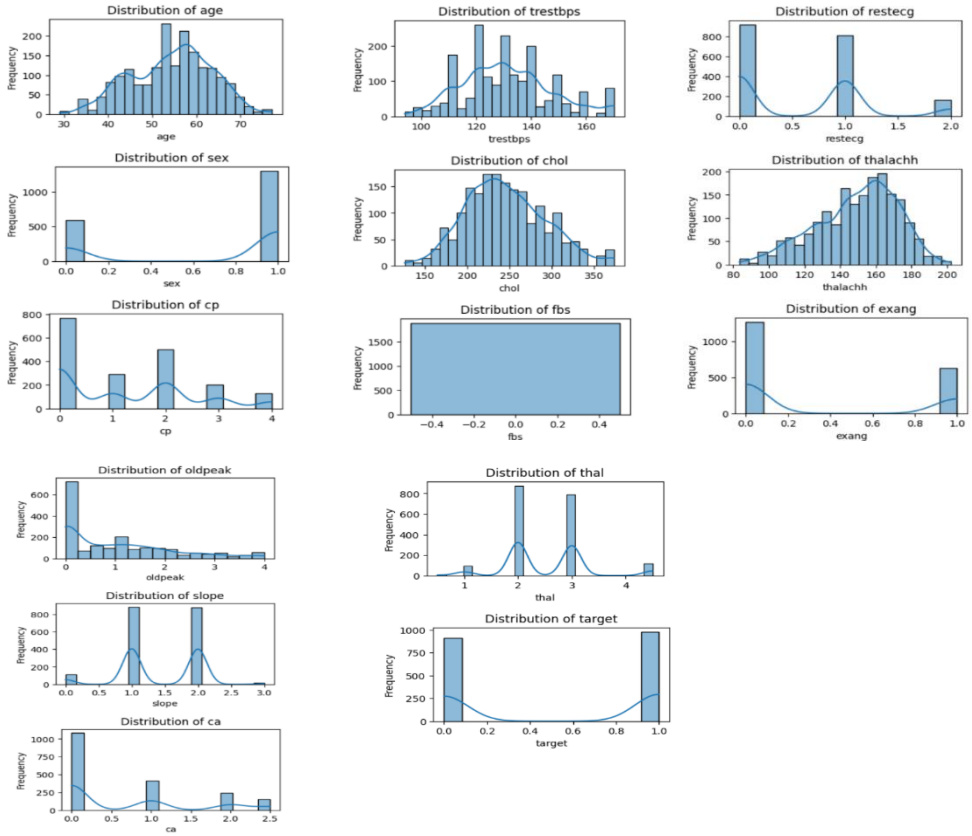
Dalam memprediksi resiko terkena penyakit jantung ini dapat dilihat data *oulier* mempengaruhi performa pada model secara negatif dan dapat memungkinkan terjadi adanya *overfitting*. Dari visualisasi tersebut, *outlier* yang terdeteksi tidak dihapus dikarenakan masih bisa diatasi dengan penanganan data *outlier* sehingga tidak termasuk dalam kesalahan. Selain itu juga penghapusan data outlier dikhawatirkan dapat menyebabkan penurunan dalam training data.

d) Distribusi Data

Memvisualisasikan distribusi data dengan memanfaatkan histogram agar lebih mudah dilihat dan dipahami.

	age	sex	cp	trestbps	chol	fbs	restecg	thalachh	exang	oldpeak	slope	ca	thal	target
count	1888.000000	1888.000000	1888.000000	1888.000000	1888.000000	1888.0	1888.000000	1888.000000	1888.000000	1888.000000	1888.000000	1888.000000	1888.00000	1888.000000
mean	0.528215	0.688559	1.279131	0.489386	0.484079	0.0	0.597458	0.556714	0.331568	0.259627	1.421610	0.675583	2.51536	0.517479
std	0.189198	0.463205	1.280877	0.217939	0.192354	0.0	0.638820	0.192913	0.470901	0.277600	0.619588	0.883531	0.78102	0.499827
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.50000	0.000000
25%	0.390625	0.000000	0.000000	0.342105	0.343434	0.0	0.000000	0.417722	0.000000	0.000000	1.000000	0.000000	2.00000	0.000000
50%	0.541667	1.000000	1.000000	0.473684	0.464646	0.0	1.000000	0.578059	0.000000	0.200000	1.000000	0.000000	2.00000	1.000000
75%	0.666667	1.000000	2.000000	0.605263	0.606061	0.0	1.000000	0.696203	1.000000	0.400000	2.000000	1.000000	3.00000	1.000000
max	1.000000	1.000000	4.000000	1.000000	1.000000	0.0	2.000000	1.000000	1.000000	1.000000	3.000000	2.500000	4.50000	1.000000

Gambar 4.5 Output Distribusi Data

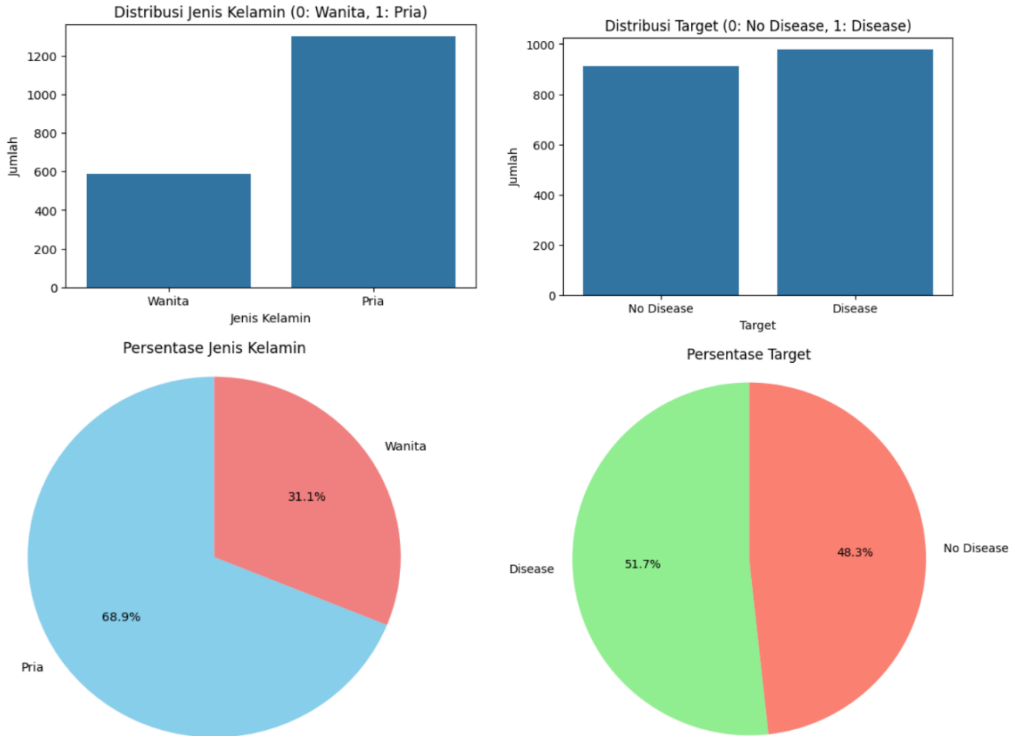


Gambar 4.6 Histogram Distribusi Data

Distribusi data pada fitur-fitur yang ditampilkan menunjukan beberapa karakteristik dari setiap kolom. Seperti *Age, Trestbs, Restecg, Sex, Chol, Thalachh, Cp, Fbs, Exang, Oldpeak, Slope, Ca, Thal* dan *Target*.

e) *Univariate Analysis*

Analisis univariate dari dataset menunjukan bebrapa temuan pada kolom penting. Untuk variabel *Sex*, distribusi menunjukan bahwa mayoritas sampel berjenis kelamin laki-laki *(Male*) sebanyak 1300 (68,9%), sementara perempuan (*Female)* hanya berjumlah 588 (31,1%). Ketidakseimbangan ini perlu diperhatikan, terumata jika variabel *Sex* digunakan dalam analisis atau model prediksi.

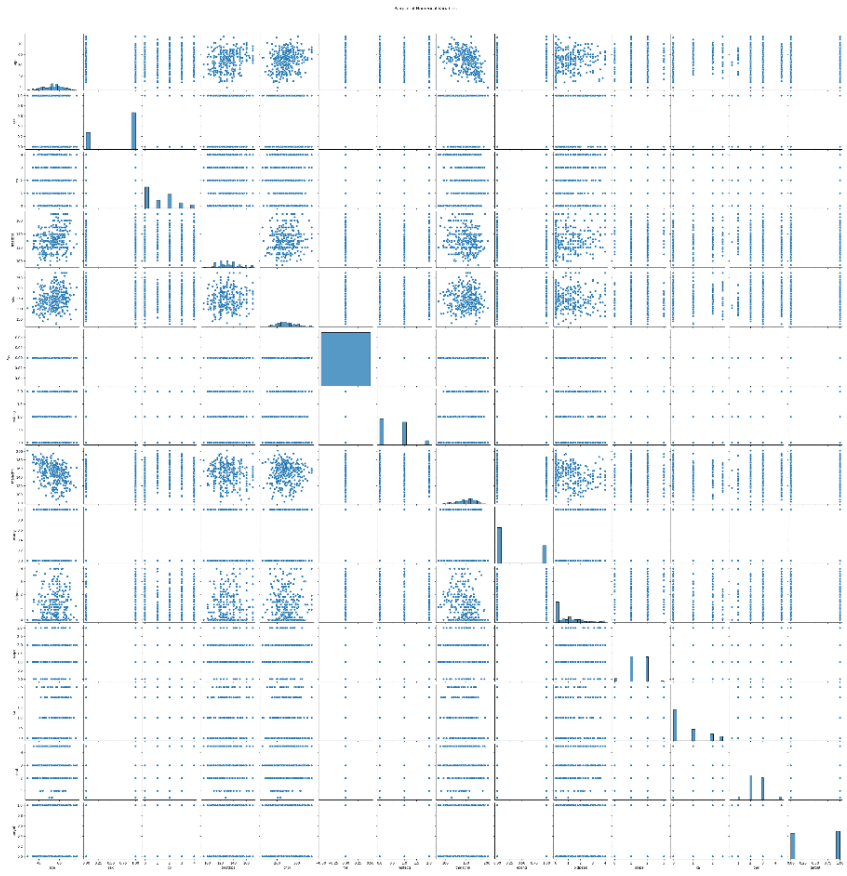


Gambar 4.7 *Univariate Analysis*

Pada variabel *Target*, sebagian besar sampel berada pada kategori “1” sebanyak 977 (51,7%), sementara kategori “0” hanya mencakup 911 (48,3%).

f) Multivariate Analysis

Pada tahap ini dimana saya menggunakan grafik pairplot ini digunakan untuk menganalisis hubungan multivariat antara variabel-variabel numerik dalam dataset, yaitu *Age, Sex, Cp, Trestbps, Chol, Fbs, Restecg, Thalachh, Exang, Oldpeak, Slope, Ca, Thal,* dan *Target.* Distribusi univariate ini menunjukan pola penyebaran data yang menggambarkan karakteristik masing-masing variabel. Variabel *Age* cenderung memiliki distribusi yang didominasi oleh sampel pada kelompok usia. Variabel *Sex* menggambarkan distribusi umur pada pasien, sementara variabel seperti *Cp, Trestbps, Chol, Fbs, Restecg, Thalachh, Exang, Oldpeak, Slope, Ca, Thal,* dan *Target* menunjukan pola distribusi yang teratur dan terfokus. Hubungan antar variabel menunjukkan berbagai pola distribusi yang teratur dan terfokus.



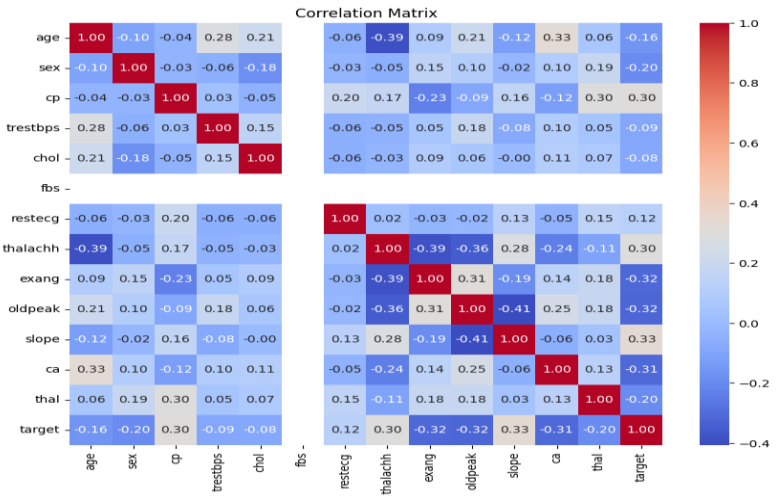
Gambar 4.8 *Multivariate Analysis*

g) *Matrix Corellation*

Matriks korelasi adalah alat statistik yang dapat digunakan untuk dapat mengukur kekuatan dan arah hubungan antara dua atau lebih variabel. Dalam kasus ini dapat diartikan menunjukan hubungan antara berbagai fitur yang berkaitan dengan data kesehatan yaitu memprediksi penyakit jantung. Dari korelasi ini terdapat beberapa nilai korelasi yaitu Rentang (-1 hingga 1), Nilai 1: Hubungan positif sempurna, Nilai -1: Hubungan negatif sempurna dan Nilai 0: Tidak ada koneksi. Analisis korelasi antara Age dan Target dapat dinyatakan korelasi -0.16 menunjukan hubungan negatif yang relatif lemah antara usia dan keberadaan penyakit jantung. Dan antara kolom Sex (jenis kelamin) dan Target korelasi yang didapat -0.20 menunjukan bahwa ada kecenderungan bahwa pria (yang berlabel 1) mungkin lebih banyak yang terkena penyakit jantung.

Sementara itu pada variabel *Chol* (Kalesterol) dapat korelasi positif dengan bebrapa variabel yang menunjukan bahwa kolesterol mungkin berhubungan dngan peningkatan risiko terkena penyakit jantung, dan ada faktor lain seperti *Thalach* (Thallasemia) yang korelasinya dapat positif) 0.30 dengan variabel Target, menunjukan bahwa detak jantung maksimum yang lebih tinggi berhubungan dengan risiko masalah jantung. Hubungan variabel *Cp* (Nyeri Dada) memiliki korelasi positif yang dikatan signifikan karena berhubungan dengan variabel *Exang* (Angina) 0.31 menunjukan bahwa ada kemungkinan tinggi mengalami angina saat berolahraga bagi mereka dengan jenis nyeri dada tertentu. *Trestbps* (Tekanan Darah) korelasi positif dengan *Chol* (Kalesterol) tidak memiliki hubungan yang kuat dengan variabel kesehatan lainnya.

Matriks korelasi ini memberikan wawasan signifikan tentang bagaimana variabel berinteraksi satu sama lain dalam konteks kesehatan jantung. Pemetaan hubungan ini penting untuk menentukan variabel mana yang mungkin berkontribusi pada risiko penyakit atau faktor lainnya dalam analisis lebih lanjut.



Gambar 4.9 *Matrix Corellation*

4.1.3 Data Preprocessing

a. Standarisasi Data

Standarisasi data adalah bagian dari tahapan transformasi/pra-pemrosesan data yang harus dilakukan sebelum tahap modeling. Langkah ini memastikan bahwa data memiliki skala yang seragam, yang sangat penting untuk algoritma machine learning.

Hasil output dari proses analisis standarisasi data yang telah dilakukan adalah sebagai berikut:

```
First few rows of X_train:
           age       sex        cp  trestbps      chol  fbs   restecg  \
1474 -0.155167  0.667065  2.120244 -0.497124  0.776598  0.0 -0.927995
1359  0.287330  0.667065  1.340894 -0.194927 -0.354760  0.0  2.238722
932   1.172323  0.667065  1.340894  0.409467  0.776598  0.0 -0.927995
618  -1.372032  0.667065  1.340894  1.013861 -0.034564  0.0 -0.927995
588   1.836068 -1.499104  0.561545 -1.282837  0.413710  0.0 -0.927995

      thalachh     exang   oldpeak     slope        ca      thal
1474 -2.369400  1.423395  0.857950  0.936587  1.464088  2.534374
1359  0.029375 -0.702546 -0.589541  0.936587  0.347536  2.534374
932   1.076113 -0.702546  0.315141 -0.672339  0.347536 -0.646232
618   1.250569 -0.702546 -0.227668  0.936587  1.464088 -0.646232
588  -0.842907 -0.702546 -0.951414  0.936587  0.347536 -0.646232

First few rows of y_train:
1474    1
1359    1
932     0
618     1
588     1
Name: target, dtype: int64
```

Gambar 4.10 Standarisasi Data

b) Splitting Data

Proses selanjutnya adalah proses membagi dataset menjadi dua bagian yaitu data training dan data testing atau data uji yang mana menggunakan rasio perbandingan 80:20. Sebelum melakukan pemisahan, langkah pertama adalah mengumpulkan seluruh dataset. Penting juga untuk dapat memahami data terlebih dahulu yaitu seperti mengecek tipe data dan dapat mengetahui fitur-fitur apa saja yang ada dalam dataset. Setelah itu lakukan tahapan EDA untuk memahami distribusi, dan lain-lain [21].

Proses ini sangat penting dilakukan untuk dapat mengevaluasi membangun performa model secara efesien. Berdasarkan rasio yang dipilih 80;20, bagian pertama 80% pertama digunakan uuntuk melatih model dan Data testing bagian 20% disimpan terpisah dan tidak boleh digunakan selama proses training atau penentuan parameter model. Data ini hanya digunakan untuk evaluasi akhir [20].

Data Training digunakan sebagai membangun dan mengoptimalkan model pada penelitian ini berjumlah 1510 record data, dan Data Testing terdapat 378 record data yang digunakan untuk mengevaluasi seberapa baik performa model dapat melakukan dan menganalisis data baru. Pembagian data yang tepat sangat penting untuk memastikan bahwa hasil evaluasi model adalah representasi akurat dari performa sebenarnya [11].

```python
# code membagi data training dan testing

import pandas as pd
import numpy as np
# Pisahkan fitur (X) dan target (y)
X = df.drop('target', axis=1)
y = df['target']

# Lakukan one-hot encoding untuk fitur kategorikal
X = pd.get_dummies(X, drop_first=True)

# Bagi data menjadi data training dan testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Lakukan standarisasi fitur numerik
from sklearn.preprocessing import StandardScaler
numerical_cols = X.select_dtypes(include=np.number).columns
scaler = StandardScaler()
X_train[numerical_cols] = scaler.fit_transform(X_train[numerical_cols])
X_test[numerical_cols] = scaler.transform(X_test[numerical_cols])
```

Gambar 4.11 Tahapan Splitting Data

Dalam python, parameter random_state digunakan agar hasil pemisahan data dapat direproduksi kembali. Setelah data dipisah, dapat menggunakan data training (X_train,y_train) untuk melatih algoritma machine learning. Setelah model selesai dillatih, hanya menggunakan data testing (X_test, y_test) untuk mengevaluasi kinerjanya. Ini memberikan gambaran yang jujur dan objektif tentang seberapa baik model dapat memprediksi data baru.

```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

print(f"Total training data : {len(X_train)}")
print(f"Total testing data : {len(X_test)}")
```

Gambar 4.12 *Splitting Data*

Hasil *output:*

```
Total training data : 1510
Total testing data : 378
```

Gambar 4.13 Hasil *Output Splitting Data*

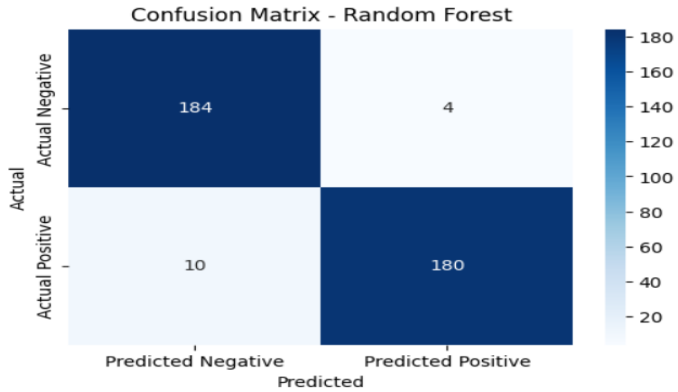
Pembagian data ini menghasilkan 1510 baris data training dan 378 baris data testing dengan masing-masing baris memiliki 13 fitur. Data yang termasuk Adalah *Age, Sex, Cp, Trestbps, Chol, Fbs, Restecg, Thalachh, Exang, Oldpeak, Slope, Ca,* dan *Thal.* Data training digunakan untuk melatih model agar mengenali pola antar fitur dan target, sedangkan data testing dipakai untuk mengevaluasi kinerja model pada data yang belum pernah dilihat sebelumnya.

4.1.4 *Model Validation*

a) *Confusion Matrix*

Confusion Matrix adalah alat yang digunakan dalam menganalisis klasifikasi untuk kinerja model atau algoritma. *Matrix* ini memberikan gambaran secara jelas tentang prediksi yang dilakukan oleh model dan dibandingkan dengan label sebenarnya.

```
Confusion Matrix (Random Forest):
[[184   4]
 [ 10 180]]
```




```
Classification Report - Random Forest:
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       188
           1       0.98      0.95      0.96       190

    accuracy                           0.96       378
   macro avg       0.96      0.96      0.96       378
weighted avg       0.96      0.96      0.96       378

Accuracy (Decision Tree): 0.96
```

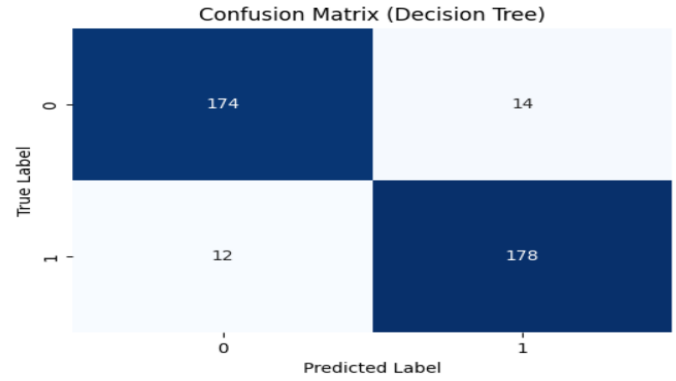
Gambar 4.14 *Confusion Matrix Random Forest*

```
Confusion Matrix (Decision Tree):
[[174  14]
 [ 12 178]]

Classification Report (Decision Tree):
              precision    recall  f1-score   support

           0       0.94      0.93      0.93       188
           1       0.93      0.94      0.93       190

    accuracy                           0.93       378
   macro avg       0.93      0.93      0.93       378
weighted avg       0.93      0.93      0.93       378
```

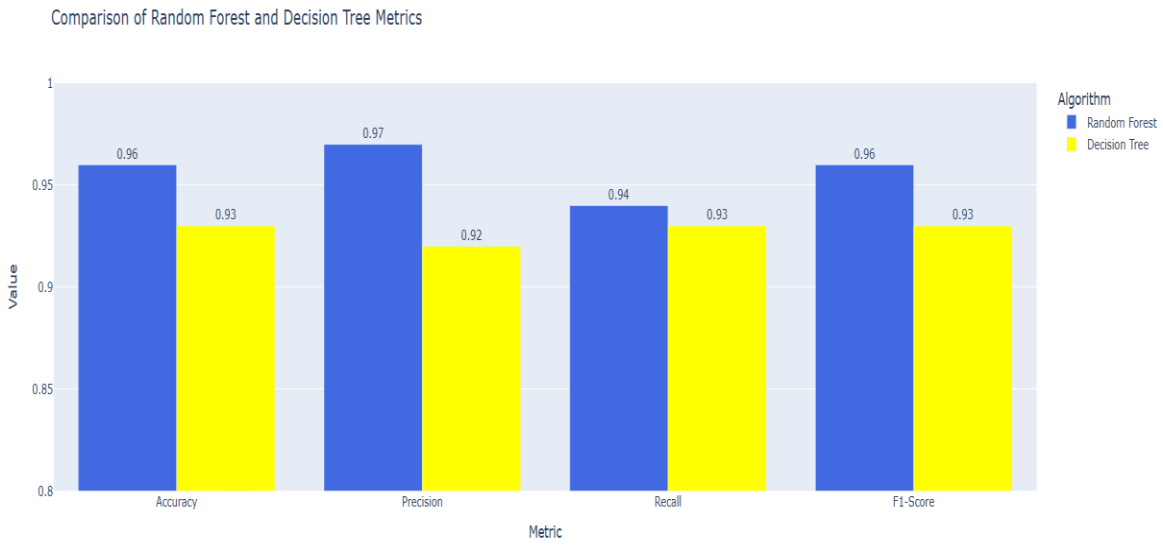



```
Accuracy (Decision Tree): 0.93
```

Gambar 4.15 *Confusion Matrix Decision Tree*

b) Visualisasi Hasil Perbandingan Algoritma

Gambar tersebut menyajikan perbandingan performa antara dua algoritma klasifikasi, yaitu *Random Forest* dan *Decision Tree*, berdasarkan empat metrik evaluasi utama: *Accuracy, Precision, Recall*, dan *F1-Score.* Berdasarkan visualisasi tersebut, terlihat bahwa algoritma *Random Forest* menunjukkan hasil yang lebih tinggi pada seluruh metrik dibandingkan Decision Tree. Pada metrik accuracy, *Random Forest* mencatatkan nilai sebesar 0.96, lebih tinggi dibandingkan *Decision Tree* yang memperoleh 0.93. Hal ini menunjukkan bahwa secara umum *Random Forest* lebih akurat dalam mengklasifikasikan data dalam memprediksi resiko terkena penyakit jantung.



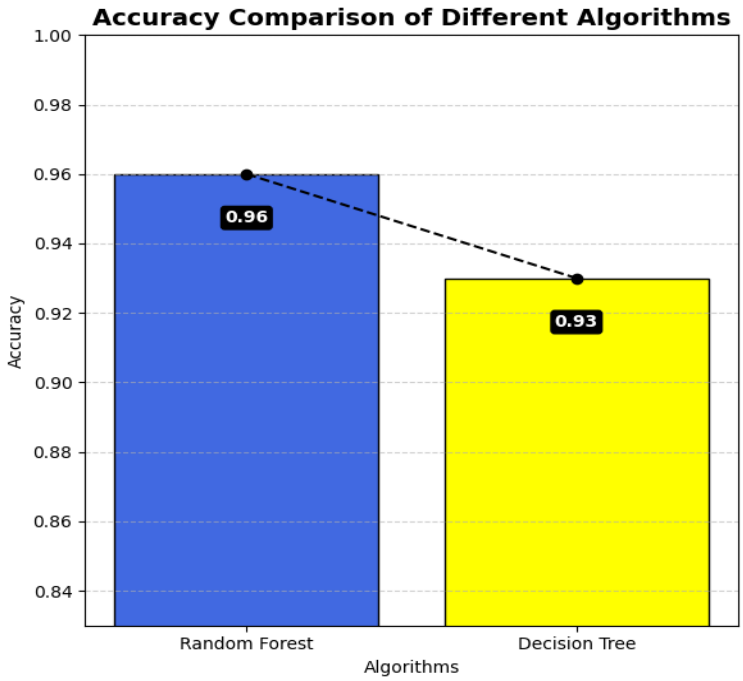
Gambar 4.16 Visualisasi Perbandingan Algoritma

Algoritma *random forest* ini adalah menggabungkan beberapa *decision tree* yang dilatih pada subset data yang berbeda dan dengan fitur yang dipilih secara acak. Dengan menggabungkan hasil dari banyak pohon keputusan, *random forest* dapat menghasilkan prediksi yang lebih akurat dan stabil dibandingkan dengan satu *decision tree* tunggal. Sedangkan algoritma *decision tre*e ini membagi data menjadi subset-subset yang semakin kecil berdasarkan nilai atribut, hingga mencapai keputusan atau prediksi pada "daun" pohon. Algoritma *decision tree* adalah salah satu algoritma *machine learning* yang paling mudah diinterpretasikan karena bentuknya yang menyerupai diagram alir.

4.1.5 *Model Evaluation*

a) *Accuracy*

Berikut adalah hasil perbandingan akurasi algoritma *Random Forest* dan *Decision Tree* untuk prediksi resiko terkena penyakit jantung:

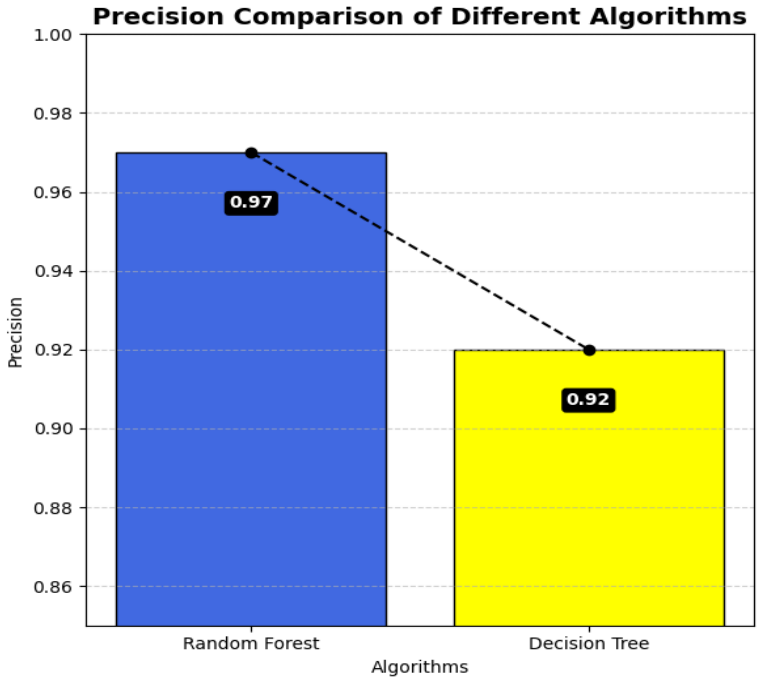


Gambar 4.17 *Accuracy*

Random Forest memiliki nilai akurasi yang lebih tinggi yaitu (0.96) dibandingkan dengan *Decision Tree* (0.93), hal ini karena algoritma *Random Forest* mampu memprediksi dengan tingkat keakuratan yang tinggi dan optimal serta algoritma ini menggabungkan prediksi dari banyak pohon keputusan, sehingga mengurangi kemungkinan adanya kesalahan yang dihasilkan oleh satu pohon tunggal. Hal ini dapat membantu dalam meningkatkan akurasi secara efektif. Sedangkan pada algoritma *Decision Tree* terdapat nilai akurasi yaitu (0.93) yang mana nilai akurasinya tidak berjauhan dengan algoritma *Random Forest*. Tetapi algortima *Decision Tree* juga memiliki keunggulan dalam tingkat analisis akurasi yaitu memberikan model yang mudah untuk dipahami, efisien dalam pengolahan data, dan fleksibel dalam penanganan berbagai jenis data. Meskipun *Random Forest* sering kali memiliki akurasi yang lebih tinggi, *Decision Tree* tetap memiliki keunggulan tersendiri dalam menganalisis data, terutama ketika interpretabilitas dan kecepatan yang sangat tinggi, tetapi setelah dianalisis algoritma *Random Forest* mungkin lebih efektif dalam konteks yang diuji karena nilai akurasinya lebih unggul dari pada algoritma *Decision Tree.*

b) *Precision*

Berikut adalah hasil perbandingan *precision* algoritma *Random Forest* dan *Decision Tree* dalam memprediksi resiko terkena penyakit jantung:



Gambar 4.18 Precision

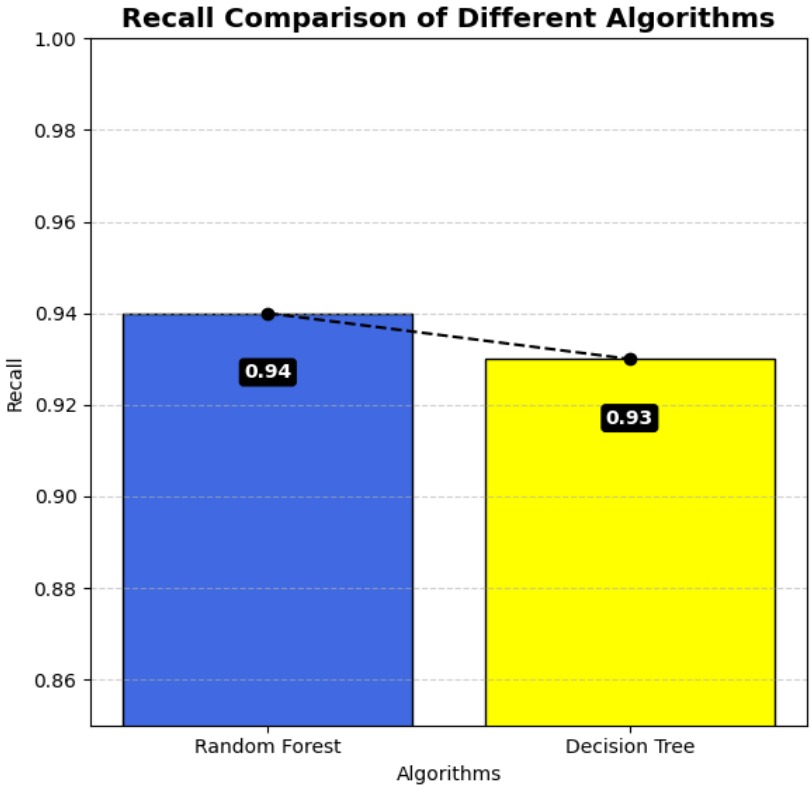
Visualisasi pada gambar 4.18 menunjukan perbandingan nilai *precision* antara dua algoritma, yaitu *Random Forest* dan *Decision Tree*. *Precision* merupakan metrik evaluasi yang mengukur seberapa akurat prediksi positif yang dilakukan oleh model menggunakan *confusion matrix*, yaitu seberapa besar proporsi prediksi positif yang benar-benar tepat. Berdasarkan grafik pada gambar 4.18, *Random Forest* memiliki nilai *precision* sebesar (0.97), sedangkan *Decision Tree* memiliki nilai *precision* sebesar (0.92).

Hal ini menunjukan bahwa *Random Forest* lebih unggul dalam menghasilkan prediksi positif yang akurat dibandingkan *Decision Tree*. Dengan nilai *precision* yang lebih tinggi, *Random Forest* cenderung lebih efektif dalam menghindari kesalahan prediksi positif palsu *(false positif)*. Oleh karena itu algoritma *Random*

Forest memiliki *precision* yang lebih tinggi (0.97) dibandingkan dengan *Decision Tree* (0.92), menunjukkan bahwa *Random Forest* lebih efektif dalam menyelesaikan tugas yang diuji.

c) *Recall*

Berikut adalah hasil perbandingan *Recall* algoritma *Random Forest* dan *Decision Tree* dalam memprediksi resiko terkena penyakit jantung:



Gambar 4.19 *Recall*

Recall merupakan metrik yang digunakan untuk mengukur kemampuan model dalam mendeteksi seluruh kasus positif yang tidak terdeteksi oleh model atau dengan kata lain, semakin rendah jumlah *(false negative)*. Berdasarkan pada gambar visualisasi 4.19 menunjukan bahwa Algoritma *Random Forest* memperoleh nilai *recall* (0.94) sedangkan *Decision Tree* mendapatkan nilai *recall* sebesar (0.93). Meskipun selisihnya terlihat kecil, hal ini menunjukan

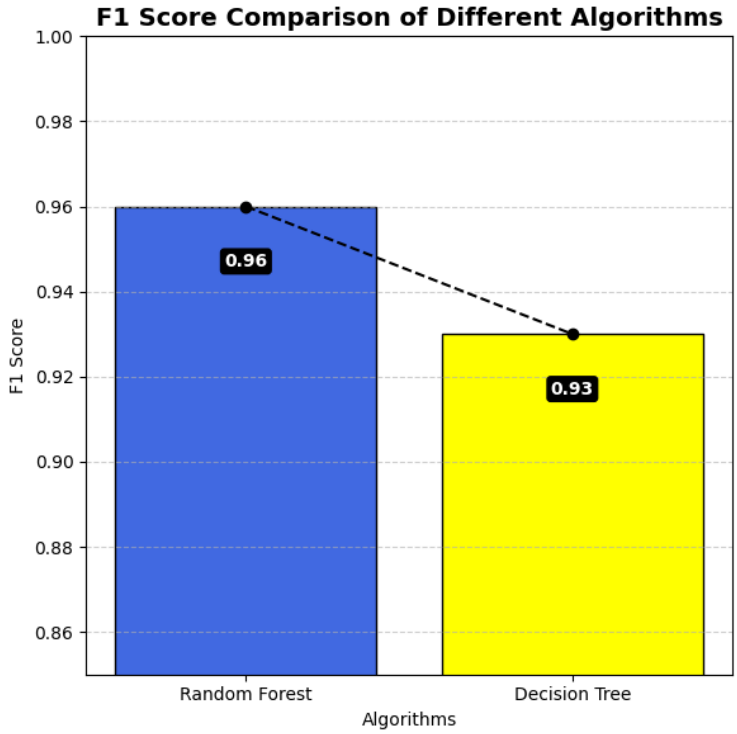
bahwa *Random Forest* lebih mampu mengenali seluruh kasus positif dibandingkan algoritma *Decision Tree*.

Perbedaan ini terjadi karena adanya cara kerja kedua algoritma tersebut. *Decision Tree* membangun satu pohon keputusan tunggal berdasarkan pembagian data disetiap node. Struktur pohon ini bersifat *deterministic* dan hanya mengandalkan satu jalur logika yang terbentuk dari pelatihan data. Akibatnya, *Decision Tree* cenderung *overfitting* terhadap data *training*, terutama jika data mengandung *noise* atau tidak seimbang. *Overfitting* ini membuat *Decision Tree* tidak mampu mengenali pola umum pada data *testing*, sehingga berisiko gagal mendeteksi sebagian kasus positif yang sebenarnya.

Disisi lain, *Random Forest* bekerja dengan membangun banyak pohon keputusan dari sampel data yang berbeda-beda, lalu menggabungkan hasil prediksi sari setiap pohon melalui voting mayoritas. Proses ini membuat *Random Forest* lebih tahan terhadap *overfitting*, lebih stabil dalam menghadapi data yang kompleks, dan lebih akurat dalam mengidentifikasi seluruh data positif. Karena banyaknya pohon yang saling melengkapi dalam membuat keputusan, kemungkinan model yang terlewatkan kasus positif menjadi sangat kecil. Hal inilah yang menjadikan nilai *recall* pada *Random Forest* lebih tinggi dibandingkan *Decision Tree*. Algoritma *Random Forest* memiliki nilai *recall* yang sedikit lebih tinggi dibandingkan dengan algoritma *Decision Tree.* Berdasarkan cara kerjanya, *Random Forest* lebih unggul dalam mengenali data positif secara menyeluruh karena pendekatan ensemble yang dimilikinya membuat model lebih akurat dalam menentukan nilai *recall* untuk mengevaluasi cakupan suatu model dalam memprediksi kelas tertentu.

d) *F1 Score*

Berikut adalah hasil perbandingan *F1 Score* algoritma *Random Forest* dan *Decision Tree* untuk prediksi resiko terkena penyakit jantung:



Gambar 4.20 *F1-Score*

Berdasarkan pada gambar 4.20 menunjukan perbandingan nilai *F1-Score* antara algoritma *Random Forest* dan *Decision Tree*, dimana *Random Forest* memperoleh nilai *F1-Score* sebesar (0.96) dan *Decision Tree* sebesar (0.93). *F1-Score* merupakan metrik yang menggabungkan *precision* dan *recall* ke dalam satu nilai tunggal menggunakan rata-rata, sehingga metrik ini sangat penting untuk memahami klasifikasi model, terutama pada data yang tidak seimbang.

Nilai *F1-Score* yang tinggi menunjukan bahwa model mampu menghasilkan prediksi yang tidak hanya akurat dalam mengidentifikasi data positif (precision tinggi), tetapi juga mampu menangkap sebagian besar data positif yang ada (*recall* tinggi). Perbedaan nilai *F1-Score* antara *Random Forest* dan *Decision Tree* terjadi karena perbedaan cara kerja masing-masing algoritma. *Decision Tree* hanya membangun satu pohon keputusan tunggal berdasarkan struktur

pembagian data terbaik, sehingga model sangat bergantung pada bentuk data training dan lebih mudah mengalami *overfitting*.
Akibatnya, performa model bisa menurun saat digunakan pada data baru karena tidak mampu menangkap variasi pola yang lebih luas. Sebaliknya, *Random Forest* merupakan ansambel algoritma yang membangun banyak pohon keputusan secara acak dari subset data dan fitur, lalu menggabungkan hasilnya melalui voting mayoritas.

Pendekatan ini membuat *Random Forest* lebih stabil, tahan terhadap *overfitting*, dan memiliki kemampuan generalisasi yang lebih baik. Karena *Random Forest* menggabungkan banyak pohon, ia mampu menyeimbangkan antara *precision* dan *recall* secara lebih konsisten, yang pada akhirnya menghasilkan nilai *F1-Score* yang lebih tinggi. Oleh karena itu, dapat disimpulkan bahwa *Random Forest* memiliki kinerja keseluruhan yang lebih baik dibandingkan algoritma *Decision Tree*.